UIUCDCS-R-77-903

UILU-ENG 77 1762

GENERATING TREES AND OTHER COMBINATORIAL OBJECTS LEXICOGRAPHICALLY

By

S. Zaks and D. Richards

November 1977

DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN · URBANA, ILLINOIS

GENERATING TREES AND OTHER COMBINATORIAL OBJECTS LEXICOGRAPHICALLY

by

S. Zaks and D. Richards

Department of Computer Science
University of Illinois at Urbana-Champaign
Urbana, Illinois 61801

November 1977

---

† This work supported in part by the National Science Foundation under grant number NSF MCS 73-03408.

I. INTRODUCTION

Motivated by the problem of generating, ranking and unranking all k-ary trees with n nodes, we solved it for the more general case of all trees with n_i nodes having k_i sons each, $i = 1, 2, \ldots, t$, and $n_0 + 1$ leaves (hence $n_0 = \Sigma_1^t(k_i - 1)n_i$).

We establish a 1-1 correspondence between those trees and the integer sequences $a_1\ a_2 \ldots a_n$, $n = \Sigma_0^t n_i$, which have n_i occurrences of k_i for $i = 1, 2, \ldots, t$, and n_0 0's, such that in each prefix $a_1\ a_2 \ldots a_\ell$, $1 \leq \ell \leq n$, the number of 0's is not greater than $\Sigma_1^t(k_i - 1) \cdot$ (number of k_i's in the prefix).

It turns out that there is also a 1-1 correspondence between these sequences and the lattice paths $L = L_0\ L_1 \ldots L_n$ in the $(t + 1)$-dimensional space, from the point $(n_0, n_1, \ldots n_t)$ to the origin $(0, 0, \ldots, 0)$, which do not go below the hyperplane $x_0 = \Sigma_1^t(k_i - 1)x_i$.† These correspondences will be shown in section 2.

The algorithm, which lexicographically generates a modified version of the above sequences, is discussed in section 3. The ranking and unranking procedures are the subject of section 4.

---

† We assume that each step in the path is directed towards the origin, and we use this assumption throughout this paper.

II. TREES, SEQUENCES AND PATHS

Ordering Of Trees

We will deal solely with ordered trees (or planted planar trees). We will follow the conventions in [5]. Let $K = (k_0, \ldots, k_t)$ and $N = (n_0, \ldots, n_t)$ be $(t+1)$-tuples of non-negative integers, such that $k_t > k_{t-1} > \ldots > k_0 = 0$ and $n_0 = \sum_1^t (k_i - 1)n_i$. We are concerned with the set of trees $T(K,N)$ where each tree has n_i nodes with k_i sons each for $1 \leq i \leq t$, and for convenience, there are $n_0 + 1$ nodes with 0 sons, i.e. leaves. If $t = 1$, then we have regular[†] k_1-ary trees with n_1 internal nodes.

There are two common ways to order k-ary trees found in the current literature which we generalize to the set $T(K,N)$. Let $|T|$ be the number of nodes in tree T, r_T be the degree of its root, and T_i be the subtree rooted at the i^{th} son of the root of T.

The ordering given in [4,5] for binary trees and in [9] for k-ary trees can be generalized to arbitrary trees as A-order as follows:

> Two trees, T and T', are in A-order, $T < T'$, if
>
> 1) $|T| < |T'|$ or
>
> 2) $|T| = |T'|$ and for some i, $1 \leq i \leq r_T$, we have
>
> a) $T_j = T'_j$ for $j = 1, 2, \ldots, i-1$ and
>
> b) $T_i < T'_i$

Let p_T be the sequence formed by consecutively numbering the nodes (by traversing T) in post-order and reading them in pre-order. Two trees, T and T', are in A-order if p_T is lexicographically less than $p_{T'}$. For binary trees in-order is interchangeable with post-order and is used in [4]. The proof of this correspondence is analogous to the proof used in the binary case.

A second ordering is given in [12,13] for k-ary trees which we generalize to arbitrary trees as B-order as follows:

[†] In a regular k-ary tree, each internal node has exactly k sons.

Two trees, T and T', are in B-order, $T < T'$, if

1) $r_T < r_{T'}$ or

2) $r_T = r_{T'}$ and for some i, $1 \leq i \leq r_T$ we have

   a) $T_j = T'_j$ for $j = 1, 2, \ldots, i - 1$ and

   b) $T_i < T'_i$

Later we give our interpretation of B-order using sequences. The best known algorithms for ranking and unranking trees are considerably more efficient when the trees are in B-order. Therefore, in this paper, we will only be concerned with generation in B-order.

Tree Sequences and Lexicographic Ordering

Define A(K,N) to be the set of integer sequences $a = a_1\ a_2 \ldots a_n$ that have n_i occurrences of the integer k_i and possesses the dominating property. A sequence, a, has the dominating property if the number of 0's is not greater than $\Sigma_1^t(k_i - 1)\cdot$(number of k_i's) for every prefix $a_1\ a_2 \ldots a_\ell$, $1 \leq \ell \leq n$. The following theorem was proved in [1] (using the palindromes of our sequences).

Theorem 1: There is a 1-1 correspondence between T(K,N) and A(K,N).

This correspondence is simple to understand and use. Given a tree, T, construct the sequence a_T by labeling each node with its number of sons and reading the labels in pre-order. The last node is not read since it is always a leaf, and its omission simplifies matters. This maps a tree to a sequence. The inverse mapping is accomplished by building a tree node by node from the sequence a_T. Begin by creating a root with degree a_1 and position a pointer there. In general, process a_i by creating a new son of the node v currently pointed to and move the pointer from v to it. If v has its requisite number of sons, backtrack to v's father.

The dominating criterion arises naturally since a tree in T(K,N) has

$\Sigma_1^t(k_i - 1)n_i + 1$ leaves. If the criterion was violated, it would indicate the existence of a completed tree which does not arise since the final leaf is omitted. The property can be written as $\Sigma_1^{\ell}(a_i - 1) \geq 0$, since the sum of the negative terms is the number of 0's, or more succinctly as

$$\Sigma_1^{\ell} a_{\ell} \geq \ell.$$

This has a simple interpretation: there are not more nodes than the collective number of sons.

Pre-order search of trees shares with other search methods the property that it inspects all of a node's ancestors before inspecting that node. This is enough to insure the sequences read are in A(K,N). Breadth-first search is such a search method and is used in [1,3]. We will use pre-order exclusively because it simplifies the generation procedure.

Similar sequences have been studied extensively in relationship to the "ballot problem" (for example [12]). They have been related to binary trees in [2] and were independently given for k-ary trees in [3,13,14]. An overview of such sequences is found in [6].

It is vital to note that $T < T'$ (i.e. in B-order) if a_T is lexicographically less than $a_{T'}$. This follows from the definitions of a_T and B-order. Note that r_T and $r_{T'}$ are equal to a_1 and $a_{1'}$ respectively. Therefore, if $r_T < r_{T'}$, then a_T precedes $a_{T'}$. If $r_T = r_{T'}$ and the first $i - 1$ subtrees are equal, then the corresponding prefixes of a_T and $a_{T'}$ are equal, since a_T is formed in a pre-order fashion. Then the argument recurs on T_i and T_i'. Therefore, if we generate the sequences of A(K,N) lexicographically, we will generate the trees of T(K,N) in B-order.

In arranging the sequences of A(K,N) in lexicographic order, as normally defined, only the relative values of the k_i's are needed. Therefore, since $k_{i+1} > k_i$ and $k_0 = 0$, we find it convenient to map the sequences of A(K,N) to sequences $b = b_1\ b_2\ \ldots\ b_n$, where $b_i = j$ if $a_i = k_j$. More formally,

b is an element of $B(K,N)$ if it contains n_i occurrences of the integer i and for every subsequence $b_1\ b_2 \ldots b_\ell$, $\Sigma_1^\ell k_{b_j} \geq \ell$. There is obviously a one-to-one correspondence between $A(K,N)$ and $B(K,N)$ that preserves the lexicographic ordering.

We note that there is also a correspondence between ordered forests and such sequences. An ordered forest consists of ordered trees which are in turn ordered. Define $F(K,N)$ in the same way as $T(K,N)$ except that $n_0 = \Sigma_1^t(k_i - 1)n_i + (f - 1)$, i.e. $F(K,N) = T(K,N)$ if $f = 1$. $A(K,N)$ is defined analogously. If we introduce a new node v of degree f and connect it to the roots of the f ordered trees we create one ordered tree. The correspondence is easily seen if we prefix the sequence a, $a \in A(K,N)$, with the integer f to get a sequence corresponding to the tree with root v and note that this sequence now has the dominating property. Our generation and ranking procedures will work identically on sequences from $F(K,N)$ as from $T(K,N)$, but we use $T(K,N)$ in our discussion for clarity.

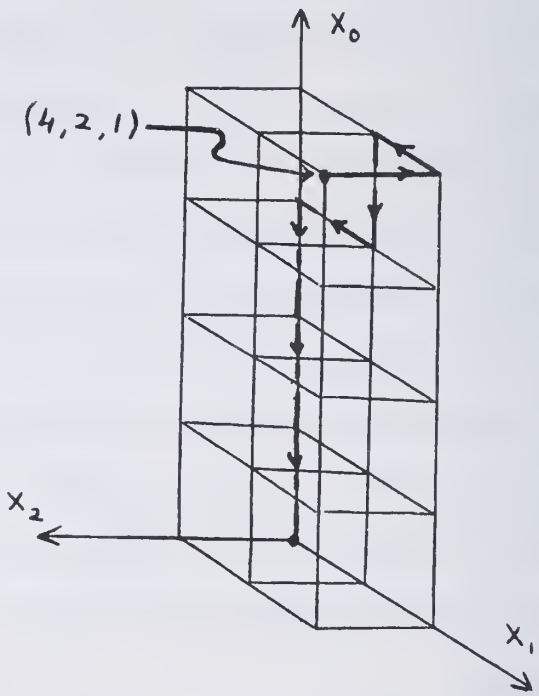
Lattice Paths

Corresponding to these sequences are lattice paths within a bounded region of $(t + 1)$-dimensional space from the point $(n_0, n_1, \ldots, n_t)$ to the origin. Each step of the path is one unit towards the origin parallel to some i^{th} dimensional axis, and the path may not go below the hyperplane $x_0 = \Sigma_1^t(k_i - 1)x_i$. Let $P(K,N)$ be this set of paths.

In [1] it is shown that there is one-to-one correspondence between $B(K,N)$ and $P(K,N)$. To map a path to a sequence, let b_j be i if the j^{th} step of the path is parallel to the i^{th} dimensional axis. The inverse mapping follows immediately. More formally, an element of $P(K,N)$ is a sequence of lattice points $L_0\ L_1 \ldots L_n$ where $L_0 = (n_0, n_1, \ldots, n_t)$, $L_N = (0, 0, \ldots 0)$ and if $b_i = j$ and $L_{i-1} = (x_0, x_1, \ldots x_t)$ then $L_i = (x_0, \ldots x_{j-1}, x_j - 1, x_{j+1}, \ldots x_t)$.

Example

As an example, consider the tree in Figure 1 which is an element of T(K,N), where K = (0,2,3) and N = (4,2,1). The corresponding elements from A(K,N), B(K,N) and P(K,N) are given.



(3 2 0 2 0 0 0) ε A(K,N)

(2 1 0 1 0 0 0) ε B(K,N)

Figure 1

III. GENERATION OF TREES

In the preceding section we described a mapping from B(K,N) to A(K,N) and from A(K,N) to T(K,N). We also showed that the lexicographic order in B(K,N) corresponds to B-order in T(K,N). To generate the next tree after a given tree T we produce its corresponding sequence $b \in B(K,N)$ and generate the lexicographically next sequence b' and map it to T', the next tree.

All the sequences of B(K,N) are permutations of each other. If it were not for the condition that each sequence should have the dominating property, it would be a simple matter of generating permutations in lexicographic order which has been well studied [8]. However, we have chosen one such algorithm and adapted it. In its original form, it can be described as follows: scan the permutation $c_1\ c_2 \ldots c_n$ of {1, 2, ... n}, from the right until the first occurrence of $c_i < c_{i+1}$. Substitute c_i with the least c_j such that $c_j > c_i$ and $j > i$ and append after it the first permutation of $\{c_i, c_{i+1}, \ldots, c_n\} - \{c_j\}$ in the ordering. (This is done efficiently by a single exchange and subsequence reversal.)

Similarly, we scan from the right for the first $b_i < b_{i+1}$ and substitute b_i with the appropriate b_j. And again, we append the first permutation of $b^* = \{b_i, b_{i+1}, \ldots b_n\} - \{b_j\}$. If it were not for the dominating property, the first permutation of b* would be $0^{m_0}\ 1^{m_1}\ 2^{m_2} \ldots t^{m_t}$ where S^x indicates x repetitions of the sequence S, and there are m_i occurrences of i in b*. Let $d = m_0 - \Sigma_1^t m_i(k_i - 1)$. The first permutation of b* is

$$0^d(1\ 0^{k_1-1})^{m_1}\ (2\ 0^{k_2-1})^{m_2} \ldots (t\ 0^{k_t-1})^{m_t}.$$

To show this, note that the dominating property can be rewritten as $\Sigma_{n-\ell+1}^{n} k_{b_i} \leq \ell$. Therefore if the first permutation of b* began with d+1 0's, the property would not hold. The next character must be the smallest non-zero character of b*; otherwise, some other sequence would precede it. Say it was a 1, then at most k_1-1 0's may follow before the property is again violated. The arguments recurs, establishing the above permutation. Note that $d > 0$, since the original sequence had the dominating property and $b_i < b_j$, when b_i and b_j were interchanged.

We now state the preceding discussion as an algorithm. Note that for termination is checked before loop entry. It is easily seen to have time complexity O(n) where $n = \Sigma_0^t n_i$.

Algorithm GENERATE (b); (This algorithm generates the lexicographically next sequence after the input sequence b).

```
for j ← 1 to t do m_j ← 0;
i ← n; sum ← 0;
while b_{i-1} ≥ b_i do
     begin m_{b_i} ← m_{b_i} + 1;
          if b_i > 0 then sum ← sum + k_{b_i} - 1;
          i ← i - 1
     end;
j ← 0; ℓ ← b_{i-1} + 1;
while j = 0 do
     if m_ℓ > 0 then j ← ℓ
     else ℓ ← ℓ + 1;
m_{b_{i-1}} ← m_{b_{i-1}} + 1;
```

```
b_{i-1} ← j;
m_j ← m_j - 1;
m_0 ← m_0 - sum;
for j ← 0 to t do
    while m_j > 0 do
        begin b_i ← j; m_j ← m_j - 1; i ← i + 1;
            for ℓ ← 1 to k_ℓ - 1 do
                begin b_i ← 0; i ← i + 1 end
        end.
```

This algorithm can be stated more succinctly in the case of k-ary trees, i.e. t = 1. In [14] this is done, but the more convenient reverse-B-order was used, where the precedence relations are merely reversed. Generation and ranking are both done differently but use sequences from B(K,N).

In [9,10] k-ary trees are ranked and generated in B-order by using a mapping between k-ary trees and binary trees. In [7] binary trees are generated and ranked in B-order. They use the sequence of the level numbers (i.e. the heights) of the leaves read in in-order. The correspondence of these two methods to B-order was established in [13]. Applying the generating algorithm to B(K,N) as in the example in the end of the section, we get the following sequences:

Index	Sequence	Index	Sequence
1	1010200	12	1210000
2	1012000	13	2001010
3	1020010	14	2001100
4	1020100	15	2010010
5	1021000	16	2010100
6	1100200	17	2011000
7	1102000	18	2100010
8	1120000	19	2100100
9	1200010	20	2101000
10	1200100	21	2110000
11	1201000		

IV. RANKING AND UNRANKING

In this section we compute the function Index(L) that, given a path $L \in P(K,N)$, will compute its corresponding position in the lexicographic ordering of P(K,N); also, given an integer w, we construct the path $L \in P(K,N)$ such that Index(L) = w. As discussed earlier, the paths $L = L_0\ L_1 \ldots L_n$ in P(K,N) are those lattice paths from the point $(n_0, n_1, \ldots, n_t)$ to the origin which do not go below the hyperplane $x_0 = \Sigma_1^t (k_i - 1)x_i$.

We make use of the multinomial coefficients

$$\binom{d}{d_1, d_2, \ldots, d_\ell} = \begin{cases} 0 & \text{if any } d_i \text{ is } < 0 \\ \dfrac{d!}{d_1!\ d_2! \cdot \ldots \cdot d_\ell!} & \text{otherwise} \end{cases}$$

where $d = \Sigma_1^\ell d_i$. The multinomial coefficient has a familiar interpretation as the number of lattice paths from the point $(d_1, d_2, \ldots, d_\ell)$ to the origin. This interpretation gives a combinatorial proof of the following lemma, which is also easily proved directly from the above definition:

Lemma 1: If $d = \Sigma_1^\ell d_i$ and all d_i are integers, then

$$\binom{d}{d_1, d_2, \ldots, d_\ell} = \sum_{i=1}^{\ell} \binom{d-1}{d_1, d_2, \ldots, d_{i-1}, d_i - 1, d_{i+1}, \ldots, d_\ell}.$$

Let $C(n_0, n_1, n_2, \ldots, n_t)$ denote the number of lattice paths from the point $(n_0, n_1, \ldots, n_t)$ to the origin which do not go below the hyperplane $x_0 = \Sigma_1^t (k_i - 1)x_i$. The following theorem defines these entries recursively, and solves the recurrence relation:

Theorem 2: The solution to the recurrence relation

$$C(n_0,n_1,n_2,\ldots,n_t) = \begin{cases} 0 & n_i<0 \text{ for } i = 1,2,\ldots, \text{ or } t. \\ 1 & n_0 = n_1 = \ldots = n_t = 0 \\ 0 & n_0 = \Sigma_1^t (k_i-1)n_i - 1 \\ \sum_{j=0}^{t} C(n_0,n_1,\ldots,n_j-1,\ldots,n_t) & \text{otherwise} \end{cases}$$

is given by

$$C(n_0,n_1,n_2,\ldots,n_t) = \binom{n}{n_0,n_1,\ldots,n_t} - \sum_{i=1}^{t} (k_i-1)\binom{n}{n_0+1,n_1,\ldots,n_i-1,\ldots,n_t} \quad (*)$$

where $n = n_0 + n_1 + \ldots + n_t$.

Proof: We show that $C(n_0,n_1, \ldots , n_t)$, as given by (*), satisfies the recurrence relation and the boundary conditions. When $n_i < 0$ for $i = 1, 2, \ldots ,$ or t (*) gives the value 0 by definition. The case $n_0 = n_1 = \ldots = n_t = 0$ is taken care of in the same way. When $n_0 = \Sigma_1^t(k_i-1)n_i - 1$ and no n_i is < 0, (*) can be rewritten as

$$C(n_0,n_1,n_2,\ldots,n_t) = \frac{n!}{(n_0+1)!n_1!\ldots n_t!}\,[n_0+1 - \Sigma_1^t(k_i-1)n_i]$$

from which it is clear that $C(n_0,n_1,\ldots,n_t)$ is 0 for this case. If n_0, n_1, $\ldots$, n_t are none of the above, we prove the recurrence by induction on n. For $n = 1$, (*) is correct. We assume that it holds for any $m < n$, and take $n = n_0 + n_1 + \ldots + n_t$. By the recursive definition of $C(n_0,n_1,\ldots,n_t)$ we have

$$C(n_0,n_1,\ldots,n_t) = \sum_{j=0}^{t} C(n_0,n_1,\ldots,n -1,\ldots,n_t).$$

For each of the terms on the right, we use (*), by the induction hypothesis, and get

$$C(n_0,n_1,\ldots,n_t) = \sum_{j=0}^{t} \binom{n-1}{n_0,n_1,\ldots,n_j-1,\ldots,n_t}$$

$$- \sum_{j=0}^{t} \sum_{i=1}^{t} (k_i-1) \binom{n-1}{n_0+1,n_1,\ldots,n_i-1,\ldots,n_j-1,\ldots,n_t}$$

$$= \sum_{j=0}^{t} \binom{n-1}{n_0, n_1, \ldots, n_j - 1, \ldots, n_t}$$

$$- \sum_{i=1}^{t} (k_i - 1) \sum_{j=0}^{t} \binom{n-1}{n_0+1, n_1, \ldots, n_i - 1, \ldots, n_j - 1, \ldots, n_t}$$

which, by the previous lemma, gives

$$C(n_0, n_1, \ldots, n_t) = \binom{n}{n_0, n_1, \ldots, n_t} - \sum_{i=1}^{t} (k_i - 1) \binom{n}{n_0+1, n_1, \ldots, n_i - 1, \ldots, n_t}$$

as desired. □

This theorem has been previously solved for the $t = 1$ case: for $k_1 = 2$ it was solved in [11] and a solution for arbitrary k_1 is found in [12]. A solution for the general problem for points on the hyperplane $x_0 = \Sigma_1^t (k_i - 1)x_i$ is given in [1] using an involved generating function argument.

Given a path $L \in P(K,N)$, we find its position Index(L) in the lexicographic ordering of $P(K,N)$, as follows:

Theorem 3: Let $b = b_1\ b_2 \ldots b_n \in B(K,N)$ and the corresponding lattice path $L = L_0\ L_1 \ldots L_n \in P(K,N)$, where $L_i = (y_{i0}, y_{i1}, \ldots, y_{it})$. Then

$$\text{Index}(L) = 1 + \sum_{i=0}^{n-1} \sum_{j=0}^{b_{i+1}-1} C(y_{i0}, y_{i1}, \ldots, y_{ij} - 1, \ldots, y_{it})$$

where the $C(.,.,\ldots,.)$'s are given by Theorem 2.

Proof: By definition we know that all the sequences that begin with either of 0, 1, ... or $b_1 - 1$ will come before this sequence b, and their number is indicated by the inner summation $\sum_{j=0}^{b_1-1}$. This follows from the definitions of b_i, y_{ij} and Theorem 2.

Next, we know that all the sequences which begin with $b_1 0$, $b_1 1$, ... , or $b_1(b_2 - 1)$ will come before b, and their number is indicated by the summation $\sum_{j=0}^{b_2-1}$. The rest follows immediately by induction, following this

line of argument.

The constant 1 is added so that the indexing will begin with 1 rather than 0. □

The time complexity depends on how the $C(x_0, \ldots, x_t)$ are calculated. If storage is inexpensive or if ranking is done frequently all the values could be stored in (t+1)-dimensional array of space complexity $O(n^*)$ in time proportional to $(t+1)n^*$ where $n^* = \Pi_1^t n_i$. Using this array, however, allows each ranking to be done with time complexity $O((t+1)n)$. If ranking is done infrequently the values of the $C(x_0, \ldots, x_t)$ can be calculated as needed in time $O((t+1)n)$. This leads to a $O((t+1)^2 n^2)$ ranking procedure. Note that for most applications (t+1) will be small and independent of n. In [9] a ranking procedure for k-ary trees is given which is $O((nk)^2)$ time-bounded. The best known ranking procedures [9] for k-ary trees in A-order are $O(n^k)$.

To illustrate this procedure refer to the tree and path in our previous example. In Figure 2 the lattice points have been labeled with $C(x_0, \ldots, x_t)$. The path L corresponding to that tree gets the following rank:

$$\text{Index}(L) = 1 + 12 + 5 + 0 + 2 + 0 + 0 + 0 = 20.$$

As for the unranking procedure, we will follow Theorem 3 in a reverse order. We are given a number w, and look for a path L such that Index(L) = w. The idea is best explained by an example: suppose we want to find the 20^{th} sequence in P(K,N) where K = (0,2,3) and N = (4,2,1). Starting at the point (4,2,1) we sum up the entries in direction 0, 1, ... (see figure 2) in that order as long as we do not exceed 20 - 1 = 19. Here we take 12, which corresponds to making the first move from (4,3,1) to (4,3,0), or $b_1 = 2$. Starting from this point (4,3,0), we can sum up the entries in the directions 0, 1, ... (in that order) as long as we do not exceed 19 - 12 = 7. Here

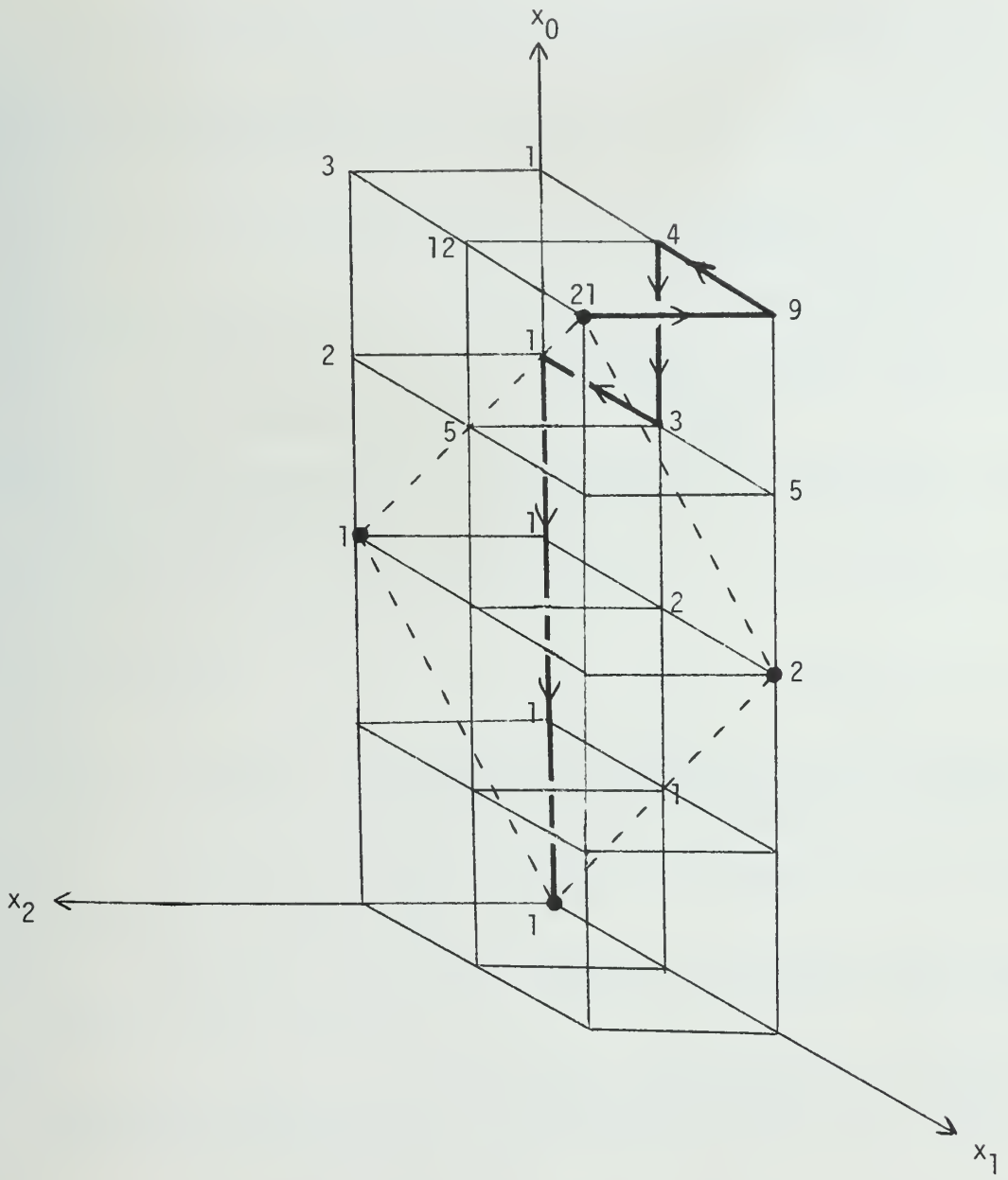


Figure 2

we take 5, which corresponds to making the next move from (4,3,0) to (3,3,0), etc. This is given more formally as algorithm UNRANK

Algorithm UNRANK(w); (This algorithm returns the b sequence corresponding to the lattice path L having rank w. The rank of the sequence beginning at L_i is always u.)

```
u ← w - 1;
(y_0,y_1,...,y_t) ← (n_0,n_1,...,n_t);
for i ← 1 to n do
    begin
        (Find the largest j such that sum of entries in
        the first j directions does not exceed u)
        j ← 0;
        sum ← 0;
        S ← C(y_0-1, y_1, ... , y_t);
        while sum + S ≤ u do
            begin sum ← sum + S; j ← j + 1;
                S ← C(y_0, ... ,y_j-1, ... , y_t)
            end;
        b_i ← j;
        y_j ← y_j - 1;
        u  ← u  - sum;
    end
```

The proof follows directly from the previous theorem. The space and time complexity considerations are the same as for the ranking procedure.

ACKNOWLEDGEMENT

The authors wish to thank Professor C. L. Liu for his valuable assistance and encouragement during this research.

REFERENCES

[1] I. Z. Chorneyko and S. G. Mohanty, "On the Enumeration of Certain Sets of Planted Plane Trees," Journal of Combinatorial Theory (B) 18, 209-221 (1975).

[2] N. G. DeBruijn and B. J. M. Morselt, "A Note on Plane Trees," Journal of Combinatorial Theory 2, 27-34 (1967).

[3] D. A. Klarner, "Correspondences Between Plane Trees and Binary Sequences," Journal of Combinatorial Theory 9, 401-411 (1970).

[4] G. D. Knott, "A Numbering System for Binary Trees," Communications of the ACM, February 1977, vol. 20, no. 2.

[5] D. E. Knuth, The Art of Computer Programming, vol. 1: Fundamental Algorithms, Addison-Wesley, 1968.

[6] R. C. Read, "The Coding of Various Kinds of Unlabeled Trees," Graph Theory and Computing, R. C. Read editor, Academic Press, 1972.

[7] F. Ruskey and T. C. Hu, "Generating Binary Trees Lexicographically," SIAM Journal on Computing, to appear.

[8] R. Sedgewick, "Permutation Generation Methods," Computing Surveys 9, 152-154 (1977).

[9] A. E. Trojanowski, "On the Ordering, Enumeration and Ranking of k-ary Trees," Technical Report UIUCDCS-R-77-850, Department of Computer Science, University of Illinois at Urbana-Champaign, February 1977.

[10] A. E. Trojanowski, "Ranking and Listing Algorithms for k-ary Trees, SIAM Journal on Computing, to appear.

[11] W. A. Whitworth, "Arrangements of m Things of One Sort and m Things of Another Sort Under Certain Conditions of Priority," Messenger of Math 8 (1878), 105-114.

[12] A. M. Yaglom and I. M. Yaglom, Challenging Mathematical Problems with Elementary Solutions, vol. 1, Combinatorial Analysis and Probability Theory, Holden-Day, 1964.

[13] S. Zaks, "Generating Binary Trees Lexicographically," Technical Report UIUCDCS-R-77-888, Department of Computer Science, University of Illinois at Urbana-Champaign, August 1977.

[14] S. Zaks, "Generating k-ary Trees Lexicographically," Technical Report UIUCDCS-R-77-901, Department of Computer Science, University of Illinois at Urbana-Champaign, November 1977.

BIBLIOGRAPHIC DATA SHEET	1. Report No. UIUCDCS-R-77-903	2.	3. Recipient's Accession No.
4. Title and Subtitle: GENERATING TREES AND OTHER COMBINATORIAL OBJECTS LEXICOGRAPHICALLY			5. Report Date: November 1977 6.
7. Author(s): S. Zaks and D. Richards			8. Performing Organization Rept. No.
9. Performing Organization Name and Address: Department of Computer Science University of Illinois at Urbana-Champaign Urbana, Illinois 61801			10. Project/Task/Work Unit No. 11. Contract/Grant No. MCS-73-03408
12. Sponsoring Organization Name and Address: National Science Foundation Washington, D.C			13. Type of Report & Period Covered 14.

15. Supplementary Notes

16. Abstracts

We show a one-to-one correspondence between all the ordered trees, that have $n_0 + 1$ leaves and n_i internal nodes with k_i sons each, for $i = 1, \ldots, t$, (hence $n_0 = \Sigma_1^t(k_i - 1)n_i$) and all the lattice paths in the $(t + 1)$-dimensional space, from the point $(n_0, n_1, \ldots, n_t)$ to the origin, which do not go below the hyperplane $x_0 = \Sigma_1^t(k_i - 1)x_i$. Procedures for generating these paths (and thus the ordered trees) are presented and the ranking and unranking procedures are derived.

17. Key Words and Document Analysis. 17a. Descriptors

ordered tree, lattice path, lexicographic order, ranking, unranking

17b. Identifiers/Open-Ended Terms

17c. COSATI Field/Group

18. Availability Statement	19. Security Class (This Report) UNCLASSIFIED	21. No. of Pages
	20. Security Class (This Page) UNCLASSIFIED	22. Price